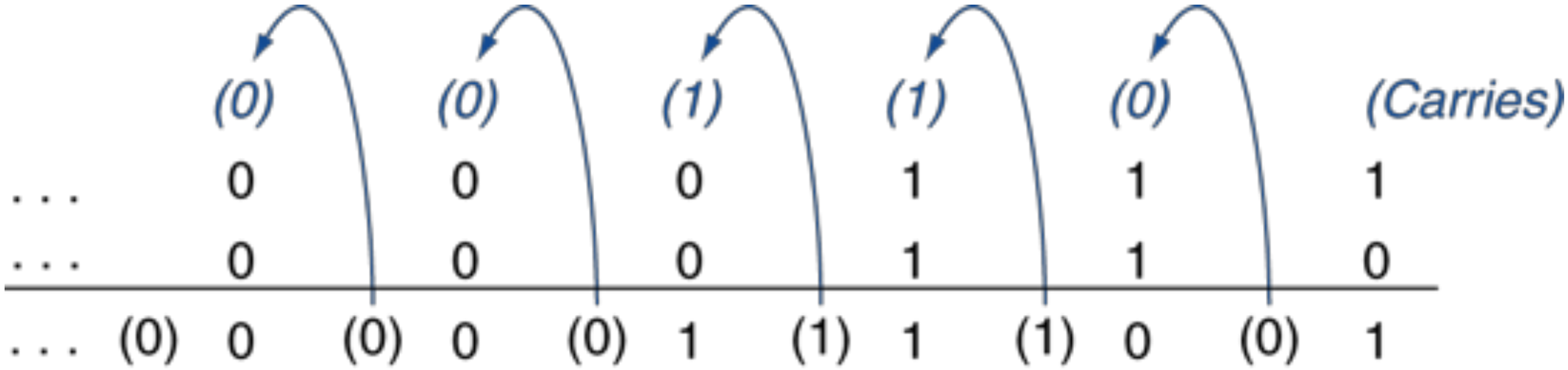
# Capitolo 3

# Aritmetica per Elaboratori

- Operazioni sugli interi
  - Addizione e sottrazione
  - Moltiplicazione e divisione
  - Gestione dell'overflow
- Numeri reali in virgola mobile
  - Rappresentazione e operazioni

# Addizione Intera

- Esempio: 7 + 6




- Overflow se il risultato è fuori dall'intervallo
  - Sommando operandi positivo e negativo, no overflow
  - Sommando operandi positivi, overflow se il bit del segno è 1
  - Sommando operandi negativi, overflow se il bit del segno è 0

# Sottrazione Intera

- Sommare la negazione del secondo operando
- Esempio: $7 - 6 = 7 + (-6)$

```
+7  0000 0000 ... 0000 0111
-6  1111 1111 ... 1111 1010
-------------------------------
+1  0000 0000 ... 0000 0001
```

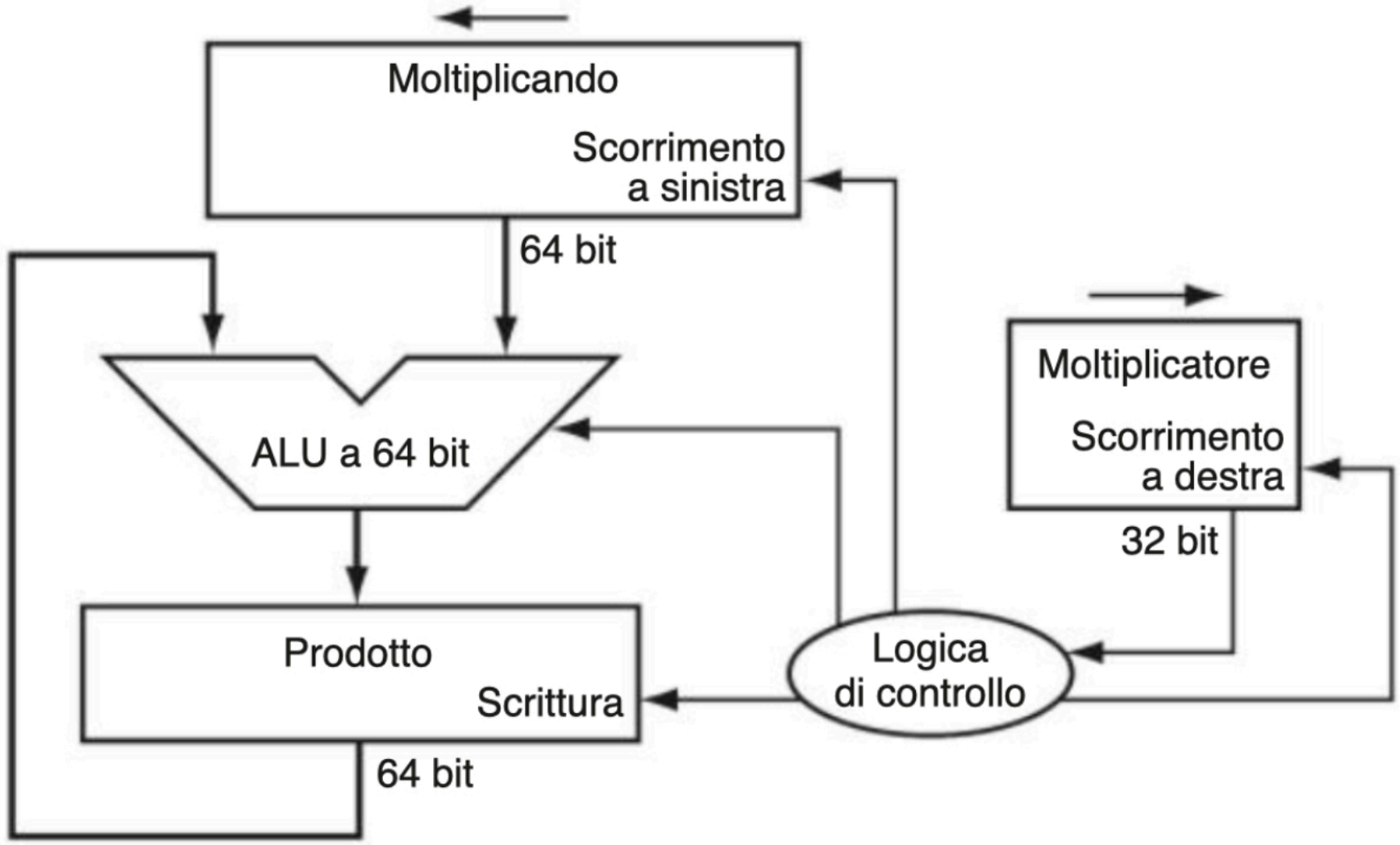
- Overflow se il risultato è fuori dall'intervallo
  - Sottraendo due operandi positivi o negativi, no overflow
  - Sottraendo un positivo da un negativo, overflow se il bit del segno è 0
  - Sottraendo un negativo da un positivo, overflow se il bit del segno è 1

# Gestire l'overflow

- Alcuni linguaggi (per esempio, il C) ignorano l'overflow
  - Usano istruzioni MIPS `addu`, `addui`, `subu`
- Altri linguaggi (per esempio, Ada e Fortran) richiedono il sollevamento di un'eccezione
  - Usano istruzioni MIPS `add`, `addi`, `sub`
  - In caso di overflow, invocano il gestore dell'eccezione
    - Salvano PC nel registro EPC (e*xception program counter*)
    - Saltano all'indirizzo predefinito del gestore
    - L'istruzione `mfc0` (*move from coprocessor register*) può leggere il contenuto di EPC, per ritornare dopo le azioni correttive

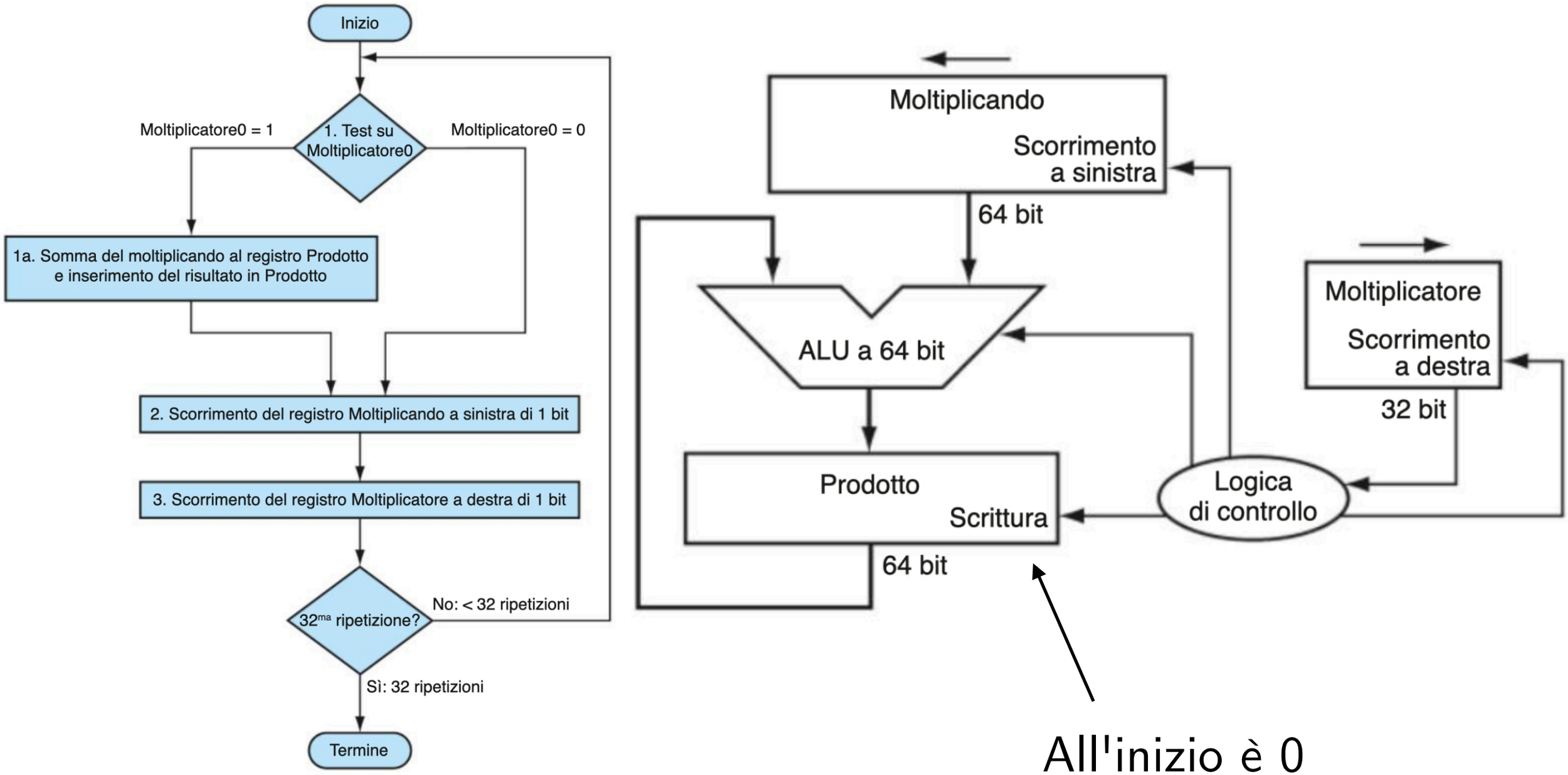
# Moltiplicazione

| | | |
|---|---|---|
| Moltiplicando | × | $1000_{dec}$ |
| Moltiplicatore | | $1001_{dec}$ |
| | | 1000 |
| | | 0000 |
| | | 0000 |
| | | 1000 |
| Prodotto | | $1001000_{dec}$ |




La lunghezza del prodotto è la somma della lunghezza degli operandi

# Hardware per la moltiplicazione



# Virgola mobile

- Rappresentazione per numeri non interi
  - Inclusi numero molto piccoli e molto grandi
- Simile alla notazione scientifica
  - $-2.34 \times 10^{56}$ ⟵ normalizzato
  - $+0.002 \times 10^{-4}$ ⟵ non normalizzato
  - $+987.02 \times 10^{9}$ ⟵ non normalizzato
- In binario
  - $\pm 1.\text{xxxxxxx}_2 \times 2^{\text{yyyy}}$
- Tipi `float` e `double` in C

# Standard virgola mobile

- Definito dallo standard IEEE Std 754-1985
- Sviluppato in risposta alla divergenza delle rappresentazioni
  - Problemi di portabilità nel codice scientifico
- Adesso adottato quasi universalmente
- Due rappresentazioni
  - Precisione singola (32-bit)
  - Precisione doppia (64-bit)

# Formato IEEE


$$x = (-1)^s \times (1 + \text{mantissa}) \times 2^{(\text{esponente} - \text{polarizzazione})}$$

- s: bit del segno (0 non-negativo, 1 negativo)
- Significando normalizzato: $1.0 \leq |\text{significando}| < 2.0$
  - Il bit prima del punto decimale è sempre uguale a 1, quindi non è necessario rappresentarlo esplicitamente
  - Il significando è la mantissa con "1." ripristinato
- Esponente: notazione polarizzata
  - esponente = esponente reale + polarizzazione
  - Assicura che l'esponente è senza segno
  - Polarizzazione 127 (precisione singola), 1023 (precisione doppia)

# Intervallo precisione singola

- Esponenti 00000000 e 11111111 riservati
- Valore più piccolo
  - Esponente: 00000001 → Esponente reale: $1 - 127 = -126$
  - Mantissa: 000...00 → Significando: 1.0
  - $\pm 1.0 \times 2^{-126} \approx \pm 1.2 \times 10^{-38}$
- Valore più grande
  - Esponente: 11111110 → Esponente reale: $254 - 127 = +127$
  - Mantissa: 111...11 → Significando $\approx 2.0$
  - $\pm 2.0 \times 2^{+127} \approx \pm 3.4 \times 10^{+38}$

# Intervallo precisione doppia

- Esponenti 000000...00 e 111111...11 riservati
- Valore più piccolo
  - Esponente: 00000000001 → Esponente reale: 1 – 1023 = –1022
  - Mantissa: 000...00 → Significando: 1.0
  - $\pm 1.0 \times 2^{-1022} \approx \pm 2.2 \times 10^{-308}$
- Valore più grande
  - Esponente: 11111111110 → Esponente reale: 2046 – 1023 = +1023
  - Mantissa: 111...11 → Significando ≈ 2.0
  - $\pm 2.0 \times 2^{+1023} \approx \pm 1.8 \times 10^{+308}$

# Somma in virgola mobile

- Si consideri un esempio su 4 cifre decimali
  - $9.999 \times 10^{1} + 1.610 \times 10^{-1}$

1. Allineare i punti decimali
   - Scorrere il numero con l'esponente più piccolo
   - $9.999 \times 10^{1} + 0.016 \times 10^{1}$
2. Sommare i significandi
   - $9.999 \times 10^{1} + 0.016 \times 10^{1} = 10.015 \times 10^{1}$
3. Normalizzare il risultato e controllare over/underflow
   - $1.0015 \times 10^{2}$
4. Arrotondare e ri-normalizzare se necessario
   - $1.002 \times 10^{2}$

# Somma in virgola mobile

- Adesso Si consideri un esempio su 4 cifre binarie
  - $1.000_2 \times 2^{-1} + -1.110_2 \times 2^{-2}$ (0.5 + –0.4375)

1. Allineare i punti binari
   - Scorrere il numero con l'esponente più piccolo
   - $1.000_2 \times 2^{-1} + -0.111_2 \times 2^{-1}$
2. Sommare i significandi
   - $1.000_2 \times 2^{-1} + -0.111_2 \times 2^{-1} = 0.001_2 \times 2^{-1}$
3. Normalizzare il risultato e controllare over/underflow
   - $1.000_2 \times 2^{-4}$, no over/underflow
4. Arrotondare e ri-normalizzare se necessario
   - $1.000_2 \times 2^{-4}$ (nessun cambiamento) $= 0.0625$

# Istruzioni FP nel MIPS

- L'hardware FP è il coprocessore 1
  - Processore aggiuntivo che estende l'ISA
- Registri FP separati
  - Precisione singole a 32 bit: `$f0, $f1, … $f31`
  - Appaiati per precisione doppia: `$f0/$f1, $f2/$f3, …`
    - La release 2 dell'ISA del MIPS supporta 32 ×registri FP a 64 bit
- Le istruzioni FP operano sono su registri FP
  - I programmi generalmente non eseguono operazioni intere su dati FP o viceversa
  - Più registri con minimo impatto sulla dimensione del codice
- Istruzioni FP di lettura e scrittura
  - `lwc1, ldc1, swc1, sdc1`
    - Per esempio, `ldc1 $f8, 32($sp)`

# Parallelismo sui dati

- La applicazioni grafiche e audio possono sfruttare l'esecuzione simultanea di operazioni su vettori
  - Esempio: sommatore a 128 bit:
    - Sedici somme a 8 bit
    - Otto somme a 16 bit
    - Quattro somme a 32 bit
- Detto anche parallelismo a livello di parola, parallelismo vettoriale, o Single Instruction, Multiple Data (SIMD)

# Note conclusive

- I bit non hanno un significato intrinseco
  - L'interpretazione dipende dalle istruzioni applicate
- Rappresentazione digitale dei numeri
  - Intervallo e precisione finiti
  - Bisogna tenerne conto nei programmi
- Aritmetica supportata nell'ISA
  - Interi con e senza segno
  - Approssimazioni in virgola mobile dei reali
- Intervallo e precisione limitati
  - Le operazioni possono causare overflow e underflow
- ISA del MIPS
  - Istruzioni principali: 54 più frequentemente usate
  - Altre istruzioni: meno frequenti